EEM 210
İŞARETLER VE SİSTEMLERİN TEMELLERİ

UYGULAMA NOTLARI

Hazırlayanlar:
Rıfat HACIOĞLU, T. Özge ÖZDİNÇ, Esen ÖZTÜRK

ZONGULDAK KARAELMAS ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ BÖLÜMÜ

ZONGULDAK, 2006

ÖNSÖZ

Hazırlanan bu uygulama notları Elektrik-Elektronik Mühendisliği ikinci sınıf öğrencilerinin İşaretler ve Sistemlerin Temelleri dersi konularına paralel olarak yapılacak çalışmaları içermektedir. Özellikle sürekli-zaman ve ayrık-zaman sistemlerinin analizi ve sentezlenmesi teknikleri ile zaman ve frekans bölgesinde işaret analiz yöntemleri üzerinde çalışılır. Ek olarak sürekli-zaman ve ayrık-zamanda filtre ve modülasyon konularında işaret işleme teknikleri tanımlanır. Ders içeriğindeki konular:

- Sürekli-zaman ve ayrık-zaman işaret ve sistemlerinin temelleri. (5 ders)
- Lineer zamanda-değişmez sistemler; Konvolüsyon (7 ders)
- Sürekli-zaman sistemlerinide Transfer Fonksiyonu gösterimi; Laplace Dönüşümü ve özellikleri (5 ders)
- Sürekli-zaman Fourier Dönüşümü ve özellikleri; Priyodik Sinyaller için Fourier Dönüşümü; filtreleme konusu (5 ders)
- Sürekli-zaman sistemlerin frekans bölgesi analizi, işaretlerin örneklenmesi; Örnekleme teoremi (5 ders)
- Ayrık-zaman Fourier Dönüşümü ve özellikleri; (5 ders)
- Z-Dönüşümü ve özellikleri; Z-Dönüşümü kullanarak ayrık-zaman sistemlerinin analizi (5 ders)
- İşaretler ve Sistemlerin frekans karakteristikleri; Log genlik ve Bode (3 ders)

Hazırlanan 11 adet uygulama ile yukarıda sıralanan konular ile ilgili çalışmalar gerçekleştirilecektir. Bu amaçla MATLAB programı kullanılacaktır. Öncelikle temel işaret-sistem analizi ile ilgili kullanılan MATLAB komutları hakkında bilgi verilecektir. Sonra da konulara paralel olarak uygulamalar ve kullanılan MATLAB komutları işlenecektir. MATLAB programlamayı en iyi öğrenme yolu bilgisayarın başına oturup verilen örnekleri gerçekleştirmek ve bu örnekler üzerinde çalışmaktır.

Öğrenciler herbir uygulama için rapor hazırlama kurallarına göre rapor hazırlamaları gerekmektedir. Bu amaçla hazırlanan raporlarda:

- Bir kapak sayfası bulunacak; dersin adı, uygulama numarası ile birlikte raporu hazırlayan grubun numarası ve hazırlayanların imzaları bulunacaktır.
- Raporun bir amacı olacak ve yapılan çalışma açıkça belirtilecektir.
- Yazılan MATLAB programı sonuçları ile birlikte verilecektir.
- Sonuç bölümünde elde edilen sonuçlar yorumlanacaktır.

B A Ş A R I L A R ...

İÇİNDEKİLER

Sayfa No

ÖNSÖZ 2
İÇİNDEKİLER 3
I.UYGULAMA: MATLAB Programı Tanıtımı 4
1.MATLAB Temelleri 4
2. Çizim 10
3. Veri Yükleme ve Saklama 12
4. FOURIER Analizi 12
5. Sürekli Zaman Sistem Analizi 15
6. Ayrık-Zaman Sistem Analizi 25
II.UYGULAMA: İşaretlerin Gösteriminde MATLAB Fonksiyonları 34
III.UYGULAMA: İşaret ve Sistemlerde Temel Kavramlar-I 38
IV.UYGULAMA: İşaret ve Sistemlerde Temel Kavramlar-II 42
V.UYGULAMA: Fark Denklemlerinin Çözümünün Bulunması 48
VI.UYGULAMA: Ayrık ve Sürekli Zaman Sistemlerde Konvolüsyon 50
VII.UYGULAMA: Polinom ve Rasyonel Fonksiyonlar, Laplace Dönüşümü 55
VIII.UYGULAMA: İşaretlerin Frekans Bileşenleri ile Gösterilmesi 58
IX.UYGULAMA: Örnekleme Teoremi ve Örtüşme Olayı 61
X.UYGULAMA: Ayrık-Zaman Fourier Dönüşümü 65
XI.UYGULAMA: Z-Dönüşümü ve Transfer Fonksiyonu ile Sistem Analizi 69

I.UYGULAMA: MATLAB Programı Tanıtımı

İlk uygulamada MATLAB programı tanıtılacak ve diğer uygulamalarda kullanılacak temel değişken tanımlama, vektör-matris oluşturma, düzyazı ve fonksiyon dosyaları oluşturma ile ilgili bilgiler verilecektir.

1.MATLAB Temelleri

A. Değişken Tanımı

İfade direkt olarak yazılarak değişkenlerin sayısal değerleri belirlenebilir. Örneğin;

a = 1+2

yazıldığında

a = 3 olarak elde edilir.

a = 1+2; şeklinde ifadenin sonuna noktalı virgül konulursa **a = 3** sonucu görülmeyecektir.

MATLAB aşağıdaki aritmetik operatörleri kullanır:

+ toplama
- çıkarma
* çarpma
/ bölme
^ güç operatörü
' transpoz

Bir değişken; aritmetik operatörlerin veya daha önceden tanımlanan değişkenlerin ya da sayıların kullanıldığı bir formülle gösterilebilir. Örneğin; a daha önceden tanımlandığı için aşağıdaki ifade geçerlidir :

b = 2*a;

Daha önceden tanımlanan bir niceliğin değerini belirlemek için niceliğin kendisi yazılır:

b

ve

b = 6

elde edilir.

Eğer bir satır ifade için yeterli değilse, üç nokta kullanılır (satır sonunda üç veya daha fazla periyodlarla) ve bir sonraki satırda tamamlanır.

c = 1+2+3+...

5+6+7;

Kullanıcı tarafından ihtiyaç duyulabilecek bazı değişkenler aşağıda verilmiştir:

i sqrt(-1)

j sqrt(-1)

pi 3.1416...

Örneğin;

y= 2*(1+4*j) yazıldığında sonucun,

y = 2.0000 + 8.0000i olduğu görülebilir.

Değişken tanımlanırken yaygın olarak kullanılan bazı fonksiyonlar:

abs	Bir sayının büyüklüğü (gerçek sayılar için absolute değer)
angle	Kompleks sayının radyan türünden açısı
cos	Kosinüs fonksiyonu
sin	Sinüs fonksiyonu
exp	Üstel fonksiyon

şeklinde verilmiştir.

Örneğin; yukarıda tanımlanan y değeriyle,

c = abs(y) eşitliği **c = 8.2462** değerini

ve

c = angle(y) eşitliği **c = 1.3258** değerini verir.

Daha önceden tanımlanan a=3 değeriyle,

c = cos(a) yazıldığında, **c = -0.9900** sonucu

ve

c = exp(a) yazıldığında ise **c = 20.0855** sonucu elde edilir.

NOT: exp kompleks sayılarda kullanılabilir. Örneğin; yukarıda tanımlanan y = 2+8i değeri kullanılarak,

c = exp(y) yazılırsa **c = -1.0751 + 7.3104i** sonucu elde edilir.

c değeri Euler formülü kullanılarak ;

c = exp(2)*cos(8) + j*exp(2)*sin(8)

şeklinde ifade edildiğinde de aynı sonucun elde edildiği görülecektir.

B. Matris Tanımı

MATLAB programlamanın temeli vektör ve matris cebri üzerine kurulmuştur. Bu nedenle, matris ve vektör işlemleri diğer yaygın hesapsal işlemlerin yapılması kadar kolaydır.

Vektörler iki şekilde tanımlanabilir. İlk yöntem rasgele elemanlar için kullanılır:

v = [1 3 5 7];

eşitliği 1, 3, 5 ve 7 elemanlarıyla 1x4'lük bir vektör oluşturur. Elemanları ayırmak için aralıklarda virgüller kullanılmaktadır. Farklı elemanlar da vektöre eklenebilir. Örneğin;

v(5) = 8;

yazılırsa

v = [1 3 5 7 8] vektörü elde edilir.

Daha önceden tanımlanan vektörler yeni bir vektör tanımlaması için kullanılabilir. Örneğin; yukarıda tanımlanan v vektörü ,

a = [9 10];

b = [v a]; şeklinde kullanıldığında ;

b = [1 3 5 7 8 9 10] vektörü oluşur.

İkinci yöntem; eşit aralıklı elemanlara sahip vektörler oluşturmak için kullanılır. Örneğin;

t = 0:.1:10; yazıldığında,

0, .1, .2, .3,...,10 elemalarından oluşan 1x101'lik bir vektör elde edilir. Ortada verilen sayı artışı belirtmektedir. Eğer iki sayı verilirse, artış 1 olarak alınır:

k = 0:10; yazıldığında,

0, 1, 2, ..., 10 elemanlarından oluşan 1x11'lik bir vektör elde edilir.

Matrisler elemanların satır satır girilmesiyle tanımlanmaktadır. Örneğin;

M = [1 2 4; 3 6 8];

$M = \begin{bmatrix} 1 & 2 & 4 \\ 3 & 6 & 8 \end{bmatrix}$ matrisini oluşturur.

Bazı özel matrisler aşağıda tanımlanmıştır :

boş matris:	**M = [ ];**
nxm'lik sıfır matrisi:	**M = zeros(n,m);**
nxm'lik bir matrisi:	**M = ones(n,m);**
nxn'lik özdeşlik matrisi:	**M = eye(n);**

Matrisin belirli bir elemanı saptanabilir. Örneğin;

M(1,2) = 5; yazıldığında,

matrisin 1.satır 2. sütununa 5 sayısı yerleşir.

Daha önceki bölümde skalarlar için tanımlanan işlemler ve fonksiyonlar aynı zamanda matris ve vektörlerde de kullanılabilir. Örneğin;

a = [1 2 3];
b = [4 5 6];

c = a + b tanımlamaları ile,

c = 5 7 9

sonucu elde edilir.

Fonksiyonlar elemen eleman uygulanır. Örneğin;

t = 0:10;
x = cos(2*t); yazıldığında,

t = 0, 1, 2, ..., 10 için cos(2t)'ye eşit elemanlı bir x vektörü oluşur.

Eleman eleman yapılması gereken işlemlerde, işlem operatöründen önce "." kullanılır. Örneğin; zamanda belirli noktalarda x(t) = tcos(t) elemanlarını içeren bir x vektörü sadece t vektörünün cos(t) vektörüyle çarpılmasıyla elde edilemez. Elemanlarını birlikte çarpmak yerine;

t = 0:10;
x = t.*cos(t);

kullanılabilir.

C. Genel Bilgi

Matlab duruma duyarlıdır bu yüzden "a" ve "A" iki farklı isimdir.

MATLAB için help yazılarak veya fonksiyon ismi ya da m-file ismi help 'ten sonra yazılarak on-line yardım alınabilir. Örneğin; **help cos** kosinüs fonksiyonu hakkında bilgi verir.

Gösterilen sayılarda kullanılan eleman sayısı sayının tam karşılığı olmak zorunda değildir. Gösterim formatı değiştirilirken; 5 desimal alanlı bilimsel notasyonlar için **format short e,** 15

önemli desimal alanlı bilimsel notasyonlar için **format long e** ve iki önemli dijiti desimalin sağına yerleştirmek için **format bank** yazılır.

who ve **whos** komutları çalışma alanında tanımlanan değişken isimlerini verir.

length(x) komutu, x vektörünün uzunluğunu ve **size(x)** komutu, x matrisinin boyutunu verir.

D. M-Dosyaları

M-dosyaları; *filename*.m şeklinde m uzantılı olarak sıradan dosyalar gibi saklanan MATLAB komutlarının makrolarıdır. Bir M-dosyası; giriş-çıkış değişkenli bir fonksiyon ya da komutların bir listesi olabilir.

Aşağıda M-dosyalarının kullanımı anlatılmaktadır. MATLAB M-dosyalarının çalışma dizininde ya da MATLAB path (yol) listesinde belirlenen başka bir dizinde saklanmasını gerektirir. Örneğin; "\MATLAB\MFILES" dizininde kullanıcı tanımlı M-dosyası saklandığı düşünülsün. Bu M-dosyasına ulaşmak için ya MATLAB komut penceresinden cd\matlab\mfiles yazılarak çalışma dizini değiştirilir ya da bu dizin yola eklenir. Yolda geçici değişiklikler MATLAB'da yazılarak yapılırken, yola sürekli eklemeler \MATLAB\matlabrc.m dosyası çalıştırılarak yapılır. Bir fonksiyonu tanımlayan M-dosyası örneği olarak , çalışma dizininizde yplusx.m isimli aşağıdaki komutları içeren bir dosya oluşturun:

```
function z = yplusx(y,x)
z = y + x;
```

MATLAB içinden yazılan aşağıdaki komutlar, bu M-dosyasının nasıl kullanıldığını gösterir:

```
x = 2;
y = 3;
z = yplusx(y,x)
```

MATLAB M-dosyaları matris veya vektör işlemlerinin kullanıldığı bir yolla yazıldığında daha verimli olur. Döngüler ve **if** ifadeleri kullanılabilir fakat hesapsal olarak yetersiz olduklarından az kullanılmalıdırlar. Aşağıda **for** komutunun kullanımı için bir örnek verilmiştir:

```
for k=1:10,
```

```
x(k) = cos(k);

end
```

Bu komutlarla, 1'den 10'a kadar olan pozitif tamsayıların kosinüsünü içeren 1x10'luk x vektörü oluşturulur. Bu işlem, **for** döngüsü yerine bir vektörün fonksiyonunun kullanıldığı aşağıdaki komutlarla daha hızlı bir şekilde yapılabilir:

```
k = 1:10;
x = cos(k);
```

if, şartlı ifadeleri tanımlamak için kullanılabilir. Örneğin;

```
if(a <= 2),
b = 1;
elseif(a >=4)
b = 2;
else
b = 3;
end
```

İfadeler arasında kullanılan karşılaştırmalar; **>=, <=, <, >, ==,** ve ~= sembolleri ile ifade edilir. Bazı M-dosyaları **input** komutuyla belirtilen kullanıcı tanımlı değişken kullanır. Örneğin, T değişkeninin farklı değerleriyle bir M-dosyası çalıştırılmak istendiği düşünülsün. M-dosyasının içindeki aşağıdaki komut satırı değeri belirler:

```
T = input('Input the value of T: ')
```

Tırnak işareti arasındaki her türlü komut, M-dosyası çalışırken ekranda gösterilir, ve kullanıcı uygun bir değer girmelidir.

2. Çizim

Kullanılan komutlar:

plot
xlabel
ylabel
title
grid
axis
stem

subplot

Çizim için en çok kullanılan komut, matrislerin ve vektörlerin doğrusal çizimlerini oluşturan **plot** komutudur. plot(t,y); x-ekseninde t vektörüne karşı, y ekseninde y vektörünü çizdirir.

plot(t,y,'option') komutu kullanılarak çizgi şeklinde ve çizim renginde değişiklikler yapılabilir. Kesiksiz çizgi (default) '-', kısa çizgi '--', nokta kısa çizgi '-.', noktalı çizgi ':' şeklinde çizgi seçenekleri vardır. Y deki noktalar birleştirilmeden bırakılabilir ve + . * o x gibi sembollerle tekrar çizgilendirilebilir. Kullanılabilen renk seçenekleri aşağıda verilmiştir:

r kırmızı

b mavi

g yeşil

w beyaz

k siyah

Örneğin; plot(t,y,'--') komutu ile kısa çizgi kullanılır; plot(t,y,'*') noktaları birleştirmeden t ve y de tanımlı bütün noktalarda * kullanır ve plot(t,y,'g') komutu ile de kesiksiz yeşil çizgi oluşturulur. Bu opsiyonlar birlikte de kullanılabilir, örneğin; plot(t,y,'g:') komutu noktalı yeşil çizgi çizdirir.

Aynı eksenlerde iki veya daha fazla grafik çizdirmek için y1'e karşı t1'i ve y2'ye karşı t2'yi çizdiren plot(t1,y1,t2,y2) komutu kullanılır.

Eksenleri isimlendirmek ve çizime başlık eklemek için;

xlabel('time (sec)')

ylabel('step response')

title('My Plot')

komutları yazılır.

Son olarak çizimin kolay okunabilmesi için grid komutu kullanılır.

grid

Sık karşılaşılan bir problem; MATLAB çizim fonksiyonlarının olması istenilenden daha farklı bir şekilde eksenleri ölçeklendirmesidir. Ancak çizim komutlarından sonra **axis** komutu kullanılarak eksenlerin otomatik olarak ölçeklendirilmesi önlenebilir.

axis([xmin xmax ymin ymax]);

xmin, xmax, ymin, ve ymax eksenler için istenilen limitlere karşı düşen değerlerdir.

Ayrık zaman işaretleri için, her bir noktayı küçük boş daire ve düz bir çizgiyle çizen **stem** komutu kullanılır. y[k]'ya karşı k'yı çizdirmek için **stem(k,y)** yazılır. **stem(k,y,'filled')** komutu ile dolu daireler elde edilebilir.

Ekranda birden fazla grafik çizdirebilmek için ekranı mxn'lik bölüme ayıran **subplot(mnp)** komutu kullanılabilir. Burada p; p=1 gibi en üst sol köşeyi içine alan grafiğin konumunu belirler. Örneğin;

subplot(211),semilogx(w,magdb);
subplot(212),semilogx(w,phase);

satırları ile üstte log-büyüklüklü bode çizimi ve altta faz çizimi elde edilir. Başlık ve eksen isimlendirmeleri **semilogx** komutu veya **plot** komutundan hemen sonra yapılabilir. Tam ekran çizim için **subplot(111)** komutu kullanılabilir.

3. Veri Yükleme ve Saklama

MATLAB kullanılırken, programdan çıkmak istenildiğinde tanımlanan vektörler ve matrisler kaydedilir. Dosyayı çalışma dizinine (directory) kaydetmek için;

save filename

yazılır. Burada "filename" kullanıcının seçimine bağlı olarak dosyaya verilecek isimdir. Kaydedilen dosyayı daha sonra çağırabilmek için ise,

load filename

kullanılır.

4. FOURIER Analizi

Kullanılan komutlar:

dft
idft
fft
ifft
contfft

dft komutu ayrık Fourier dönüşümünü hesaplamak için açık bir yöntem kullanır. Bir x vektörü tanımlanır ve DFT hesaplamak için,

X = dft(x)

komutu kullanılır. X'deki ilk eleman X(0) değerine karşı düşer. Bu fonksiyon

```
%
%  Discrete Fourier Transform
%
function Xk = dft(x)
[N,M] = size(x);
if M ~=1,   % makes sure that x is a column vector
  x = x';
  N = M;
end
Xk=zeros(N,1);
n = 0:N-1;
for k=0:N-1
  Xk(k+1) = exp(-j*2*pi*k*n/N)*x;
end
```

idft komutu ters ayrık Fourier dönüşümü hesaplamak için kullanılır. Bir X vektörü tanımlanır ve IDFT hesaplamak için,

x = idft(X)

komutu kullanılır. Elde edilen x vektöründeki ilk eleman x[0] değeridir.
Daha hızlı fakat daha az açık bir program için ayrık Fourier dönüşümü, bir sayı dizisinin Hızlı Fourier Dönüşümü'nü alan **fft** komutu kullanılarak hesaplanabilir. x vektöründe depolanan x[n] dizisinin FFT'sini hesaplamak için,

X = fft(x)

komutu kullanılır.
Bu şekilde kullanımla fft komutu ile dft komutu birbirinin yerine kullanılabilir. Daha fazla hesapsal verimlilik için x vektörünün uzunluğu 64, 128, 512, 1024, 2048, vb. 2'nin kuvvetlerine eşit olmalıdır. Uygun uzunluğu elde etmek için x vektöründe sıfırlar kullanılabilir. MATLAB bunu,

X = fft(x,N);

komutu ile otomatik olarak yapar. N değeri 2'nin kuvveti olacak şekilde tanımlanır.

x uzunluğu arttıkça FFT için daha iyi çizim elde edilir. Yayılma etkisinden dolayı, FFT'nin sadece ilk N/2 noktası anlamlı olacaktır.

ifft komutu ters Fourier dönüşümünü hesaplar:

x = ifft(X);

FFT sürekli zaman işaretinin Fourier dönüşümüne yaklaşım için de kullanılabilir. Bir x(t) sürekli zaman işareti T saniye periyodu ile örneklenir, daha sonra örneklenen işaret için DFT hesaplanır. Elde edilen genlik örneklenmelidir ve uygun frekans belirlenmelidir. Örneklenmiş sürekli zaman işaretinin Fourier dönüşümüne yaklaşan bir M-dosyası contfft.m 'den indirilebilir. Bir x vektörü örneklenmiş sürekli zaman işareti x(t) olarak tanımlansın ve T de örnekleme zamanı olsun.

[X,w] = contfft(x,T);

Fourier dönüşümü sonuçları X vektörü ve uygun frekans vektörü v'de saklanır. Bu fonksiyon

```
function [X,w] = contfft(x,T);
% CONTFFT [X,w] = contfft(x,T)
%
% Computes the Fourier transform of a continuous time signal
%  using the FFT.  The input is the sampled continuous
%  time signal x and the sampling time T.  The output is
%  the Fourier transform X(w) and the frequency vector w.
%
[n,m] = size(x);
if n<m,
  x = x';
end
Xn = fft(x);
N = length(x);
n = 0:N-1;
n(1) = eps;
X = (1-exp(-j*2*pi*n/N))./(j*2*pi*n/N/T).*Xn.';
w = 2*pi*n/N/T;
```

5. Sürekli Zaman Sistem Analizi

Not: MATLAB'ın son versiyonları bir sistemi göstermek için durum uzay modelini kullanır (sistem sys, sys = ss(A,B,C,D) olarak tanımlanır). Aşağıda listelenen komutların çoğu tercih edilen giriş argümanı olarak num ve den yerine sys'ye sahiptir.

A. Transfer Fonksiyonu Gösterimi

Kullanılan komutlar:

tf2zp
zp2tf
cloop
feedback
parallel
series

Transfer fonksiyonları MATLAB'da, pay ve payda katsayılarının vektörlerde saklanmasıyla tanımlanır.

$$H(s) = \frac{B(s)}{A(s)}$$

şeklinde bir sürekli zaman transfer fonksiyonu verilsin.

$$B(s) = b_M s^M + b_{M-1} s^{M-1} s + \ldots + b_0$$

ve

$$A(s) = a_N s^N + a_{N-1} s^{N-1} s + \ldots + a_0$$ 'dır.

B(s) ve A(s) 'nin katsayıları $num = [b_M \; b_{M-1} \ldots b_0]$ ve $den = [a_N \; a_{N-1} \ldots a_0]$ şeklinde vektörlerde gösterilir. Örneğin;

$$H(s) = \frac{2s + 3}{s^3 + 4s^2 + 5}$$

Transfer fonksiyonu için,

```
num = [2 3];
den = [1 4 0 5];
```

olarak tanımlanır.

Not: Katsayılar sıfır olsa bile vektörde bulunmalıdır.

Bir transfer fonksiyonu aynı zamanda sıfırları, kutupları ve kazancı ile de gösterilebilir:

$$H(s) = \frac{k(s-z_1)(s-z_2)\ldots(s-z_m)}{(s-p_1)(s-p_2)\ldots(s-p_n)}$$

Bir transfer fonksiyonunun sıfırlarının, kutuplarının ve kazancının pay ve payda polinomlarının katsayılarını içeren num ve den vektörlerinden bulunabilmesi için;

[z,p,k] = tf2zp(num,den)

yazılır.

Sıfırlar 'z' ile, kutuplar 'p' ile ve kazanç 'k' ile gösterilir. Z, p, k'dan pay ve payda polinomlarını bulmak için;

[num,den] = zp2tf(z,p,k)

komutu kullanılır.

Paralel, seri veya geribeslemede tek sistemlerin genel transfer fonksiyonu MATLAB yardımıyla elde edilebilir. G transfer fonksiyonu numG ve denG'de, H transfer fonksiyonu numH ve denH'de saklanır.

Genel geribesleme sistemini, tek transfer fonksiyonu (T(s) = G(s)/(1+G(s)H(s))) 'ye indirgemek için ;

[numT,denT] = feedback(numG,denG,numH,denH);

yazılır.

Birim geribesleme sistemi için, yukarıdaki algoritma uygulanmadan önce numH = 1 ve denH = 1 olarak tanımlanır. Daha sonra,

[numT,denT] = cloop(numG,denG,-1);

komutu kullanılır.

Seri sistemi tek bir transfer fonksiyonuna (T(s) = G(s)H(s)) indirgemek için;

[numT,denT] = series(numG,denG,numH,denH);

yazılır.

[numT,denT] = parallel(numG,denG,numH,denH);

komutu ile de paralel sistem tek bir transfer fonksiyonuna (T(s) = G(s) + H(s)) indirgenebilir.

B. Zaman Simülasyonları

Kullanılan komutlar:

residue
step
impulse
lsim

Bir sistemin zaman yanıtının bulunması için analitik yöntem; çıkış Y(s)'nin ters Laplace dönüşümünün alınmasını gerektirir. Bu işlemde MATLAB **residue** komutu kullanılarak Y(s)'nin kısmi kesirlere açılımı hesaplanabilir. Y(s)'nin pay ve payda katsayıları num ve den'de saklanır ve

[r,p,k] = residue(num,den)

yazılır.

Böylece kısmi kesir açılımı bilindiği zaman; y(t)'nin analitik ifadesi hesaplanabilir.

MATLAB'da özel bir girişe olan sistem yanıtını bulmak için sayısal yöntem de kullanılabilir. İlk once, transfer fonksiyonunun pay ve paydası sırasıyla num ve den'de saklanır. Basamak yanıtını çizdirmek için;

step(num,den)

yazılır.

Darbe cevabını çizdirmek için;

impulse(num,den)

kullanılır.

Herhangibir yanıta girişi bulmak **lsim** komutu kullanılabilir. Yanıtın hesaplanması istenilen saniye değerlerini içeren bir t vektörü oluşturulur. Genellikle bu;

t = a:b:c;

şeklinde yapılır. Burda a başlangıç zamanı, b zaman adımı ve c bitiş zamanıdır. Düzgün çizimler için t'de b en azından 300 eleman içerecek şekilde seçilmelidir. Giriş x zamanın bir fonksiyonu olarak tanımlanır, örneğin rampa x=t ile verilir. Daha sonra yanıt,

lsim(num,den,x,t);

ile çizdirilir.

Komutların istenilen şekilde uyarlanması için, zaman vektörü açıkca tanımlanmalıdır ve basamak yanıtı bir vektöre kaydedilmelidir. Beş-altı zaman sabiti için yanıtı simüle etmek genellikle sistem davranışını göstermek için yeterlidir. Kararlı bir sistem için zaman sabiti, p'nin en büyük gerçek kısmı gösterdiği (orjine en yakın) 1/Re(-p) ile hesaplanabilir.

Örneğin;

$$H(s) = \frac{2}{s+2}$$

şeklinde tanımlanan bir transfer fonksiyonu ele alınsın. Basamak yanıtı y aşağıdaki komutlarla hesaplanır ve çizdirilir:

```
num = 2; den = [1 2];
t = 0:3/300:3; % for a time constant of 1/2
y = step(num,den,t);
plot(t,y)
```

Darbe cevabı için sadece 'step' kelimesi yerine 'impulse' kelimesi kullanılır. Herhangibir x girişine yanıt ;

```
y = lsim(num,den,x,t);
plot(t,y)
```

komutları kullanılır.

C. Frekans Cevabı Çizimleri

Kullanılan komutlar:

freqs
bode
logspace
log10
semilogx
unwrap

Bir transfer fonksiyonunun frekans yanıtı H'ı hesaplamak için, transfer fonksiyonunun pay ve paydası num ve den vektörlerinde saklanır. Hesaplanacak frekansları içeren bir w vektörü tanımlanır. Örneğin; w = a:b:c şeklinde yapılan bir tanımlamada a en düşük frekansı, c en yüksek frekansı ve b de frekanstaki artışı göstermektedir.

H = freqs(num,den,w) komutu,

w'deki herbir frekans için frekans yanıtı değerini içeren kompleks bir H vektörü seçer.

Num ve den vektörlerinde gösterilen bir transfer fonksiyonunun Bode çizimi için;

bode(num,den)

komutu kullanılır.

İstenilen çizimin gerçekleştirilebilmesi için, ilk önce Bode çiziminin hesaplanacağı frekansları içeren w vektörü tanımlanır. w logaritmik skalada tanımlandığı için **logspace** komutu kullanılır. Örneğin; 0.1-100 aralığında değişen frekanslardaki Bode çizimi için w;

w = logspace(-1,2);

şeklinde tanımlanır. Bode çizimi için genlik ve faz bilgisi;

[mag,phase] = bode(num,den,w);

komutu ile elde edilebilir.

magdb = 20*log10(mag); komutu kullanılarak genlik desibel'e dönüştürülebilir.

Sonuçların y-ekseninin doğrusal, x- ekseninin logaritmik olduğu yarı logaritmik(semilog) skalada çizdirebilmesi için genlik çiziminde;

semilogx(w,magdb)

faz çiziminde;

semilogx(w,phase)

komutları kullanılır. Genlik çizimi istenilmeyen $\pm 2\pi$ atlamaları içerebilir. Bu atlamaları ortadan kaldırabilmek için, faz çiziminden önce **unwrap** komutu kullanılır.

semilogx(w,unwrap(phase))

D. Analog Filtre Tasarımı

Kullanılan komutlar:

buttap
cheb1ab
zp2tf
lp21p
lp2bp
lp2hp
lp2bs

MATLAB Butterworth filtre ve Type I Chebyshev filtre tasarımının da bulunduğu analog filtre tasarımı için çeşitli komutlar içerir. **buttap** ve **cheb1ab** komutları sırasıyla 1 rad/sn kesim frekanslı alçak geçiren Butterworth filtre ve Type I Chebyshev filtre tasarımı için kullanılır. n-kutuplu Butterworth filtre için;

[z,p,k] = buttap(n)

yazılır. Burada 'z' sıfırları, 'p' kutupları ve 'k' filtre kazancını göstermektedir. Geçiş bandında Rp desibel salınımlı n-kutuplu Type I Chebyshev filtre,

[z,p,k] = cheb1ab(n,Rp)

ile elde edilir. z, p ve k'dan elde edilen filtrenin pay ve payda polinomlarının bulunmasında,

[b,a] = zp2tf(z,p,k)

kullanılır. a payda katsayılarını ve b pay katsayılarını içerir.

Alçak geçiren filtreden farklı kesim frekanslı başka bir alçak geçiren filtreye, alçak geçiren filtreden yüksek geçiren filtreye, veya alçak geçiren filtreden band durduran ya da band geçiren filtreye frekans dönüşümleri MATLAB yardımıyla yapılabilir. Bu dönüşümler ya Butterworth filtre ile ya da Cbeyshev filtre ile kullanılabilir. b ve a 1 rad/sn kesim frekanslı alçak geçiren filtre transfer fonksiyonunun pay ve paydası olsun. Pay ve payda katsayıları b1 ve a1'de belirtilen Wo kesim frekanslı alçak geçiren filtre için;

[b1,a1] = lp2lp(b,a,Wo)

komutu kullanılır.

Wo kesim frekanslı yüksek geçiren filtre;

[b1,a1] = lp2hp(b,a,Wo)

ile tasarlanabilir.

Wo frekans merkezli, Bw band genişlikli band geçiren filtre için;

[b1,a1] = lp2bp(b,a,Wo,Bw)

yazılır.

[b1,a1] = lp2bs(b,a,Wo,Bw)

Komutu ile ise Wo frekansı civarında merkezlenmiş, durdurma bandı band genişliği Bw olan band durduran filtre tasarlanabilir.

E. Kontrol Tasarımı

rlocus komutu kullanılır.

G(s)H(s) = KP(s) olan geribesleme döngüsü ele alınsın. Burada, K kazancı gösterir ve P(s) denetleyicinin ve sistemin sıfır ve kutuplarını içerir. Kök yer eğrisi; kazanç değişimine bağlı olarak kapalı döngü transfer fonksiyonu köklerinin çizimidir. P(s)'nin pay ve payda katsayıları num ve den vektörlerinde saklansın. Aşağıdaki komut ile kök yer eğrisi çizdirilebilir ve hesaplanabilir:

rlocus(num,den)

0-100 aralığında değişen belirli bir K değeri için çizim;

```
K = 0:100;
r = rlocus(num,den,K);
plot(r,'.')
```

komutları kullanılarak elde edilebilir.

Grafik kompleks düzlemde, 0'dan 100'e kadar değişen K tamsayı değerleri için kapalı döngü kutup yerlerinde noktalar içerir. K tanımlanırken **K = 0:.5:100** şeklinde küçük artış değerleri kullanılırsa daha iyi sonuçlar elde edilebilir. Sonuçlanan r matrisi, K vektöründe tanımlanan bütün kazançlar için kapalı döngü kutuplarını içerir. Bu özellikle, K'nın belirli bir değeri için kapalı döngü kutuplarının hesaplanmasında yararlıdır. Kök yer eğrisi tamamiyle gerçek eksende yer alırsa **plot(r,'.')** kullanımı hatalı sonuçlar verir.

F. Durum Uzay Gösterimi

Kullanılan komutlar:

ss
step
lsim
ss2tf
tf2ss
ss2ss

MATLAB'da kullanılan,

$$\dot{x} = Ax + Bu$$
$$y = Cx$$

şeklindeki durum uzay gösteriminde, x nx1'lik vektör, u mx1'lik, y px1'lik, A nxn'lik, B nxm'lik, ve C pxn'liktir. Çeşitli girişlere olan sistem yanıtı, transfer fonksiyonu gösterimleri için kullanılan; **step**, **impulse**, ve **lsim** komutları ile bulunabilir. Pay ve payda vektörleri yerine A, B, C, ve D matrisleri kullanılmaktadır. Sistem;

sys = ss(A,B,C,D);

komutu ile modellenebilir.

Daha sonra, sys diğer komutlar için giriş elemanı olarak kullanılabilir. Örneğin; basamak yanıtı aşağıdaki komutlardan birinin yazılmasıyla elde edilebilir:

[y,t,x] = step(A,B,C,D);

[y,t,x] = step(sys);

Durumlar x'de, çıkışlar y'de, otomatik olarak üretilen zaman vektörü t'de gösterilir. x ve y satırları, t'de herbir zaman noktası için durumları ve çıkışları içerir. x'in herbir sütunu bir durumu ifade eder. Örneğin; zamana göre ikinci durum çizimi için;

plot(t,x(:,2))

kullanılır. Herhangi bir giriş yanıtının bulunması için veya başlangıç koşullarına yanıtın bulunması için **lsim** kullanılır. Bir t zaman vektörü ve t ile aynı satır sayısına, giriş sayısı ile aynı sütun sayısına sahip giriş matrisi u tanımlansın. x0 başlangıç durum vektörüdür. Komut;

[y,x] = lsim(A,B,C,D,u,t,x0) or [y,x] = lsim(sys,u,t,x0)

şeklinde verilir.

Tek-giriş/tek-çıkış (SISO) sistem için transfer fonksiyonu ;

[num,den] = ss2tf(A,B,C,D);

komutu kullanılarak bulunabilir. Burada, pay katsayıları 'num'da ve payda katsayıları 'den'de saklanır.

P dönüşüm matrisi verilsin, **ss2ss** fonksiyonu benzerlik dönüşümünü gerçekleştirecektir. Durum uzay modeli A, B, C, D ile ve dönüşüm matrisi P ile gösterilir.

[Abar,Bbar,Cbar,Dbar]=ss2ss(A,B,C,D,P) or [Abar,Bbar,Cbar,Dbar] = ss2ss(sys,P)

$$\dot{x} = \overline{A}x + \overline{B}u \qquad \overline{A} = PAP^{-1}, \quad \overline{B} = PB, \quad \overline{C} = CP^{-1}$$
$$y = \overline{C}x + \overline{D}u$$

şeklinde ifade edilen durum uzay gösteriminden elde edilen z=Px benzerlik dönüşümünü gerçekleştirir.

6. Ayrık-Zaman Sistem Analizi

A. Konvolüsyon

Kullanılan komutlar:

conv , deconv

x[n]*h[n] ayrık zaman konvolüsyonu için x[n] ve h[n] dizilerindeki elemanlarla x ve h vektörleri tanımlanır. Daha sonra,

y = conv(x,h)

komutu kullanılır. Bu komut x'teki ilk elemanı ve h'daki n=0'a denk düşen ilk elemanı alır böylece elde edilen çıkış vektöründeki ilk eleman n=0'a denk düşer. Eğer durum böyle olmazsa, çıkış vektörü doğru bir şekilde hesaplanır ancak dizin ayarlanmalıdır. Örneğin;

```
x = [1 1 1 1 1];
h = [0 1 2 3];
y = conv(x,h);
```

satırları ile y = [0 1 3 6 6 6 5 3] elde edilir. x yukarıda tanımlanan şekilde dizinlenirse, y[0] = 0, y[1] = 1, ... olur. Örneğin; h'deki ilk eleman n=-2'ye ve x'teki ilk eleman da n=-3'e denk düşerse, y'deki ilk eleman n=-5'e denk düşer.

Sonsuz süreçli sinyallerin konvolüsyonu hesaplanırken dikkatli olunmalıdır. x vektörü q uzunluklu ve h vektörü r uzunluklu ise, min(q,r) uzunluğuna sahip olabilmesi için y vektörü kısaltılmalıdır.

conv komutu polinomların çarpımında da kullanılabilir: a(s) katsayıları a vektöründe ve b(s) katsayıları b vektöründe verilsin, a(s)b(s) polinomunun katsayıları **ab = conv(a,b)** ile tanımlanan vektörün elemanları olarak bulunabilir.

deconv komutu konvolüsyon işleminin tersidir. Katsayıları a ve b vektörlerinde belirtilen a(s) ve b(s) verildiğinde, c(s) = b(s)/a(s) katsayıları **c = deconv(b,a)** komutu kullanılarak bulunur.

B. Transfer Fonksiyonu Gösterimi

Bir ayrık zaman transfer fonksiyonu için, katsayılar z'nin azalan kuvvetlerinde veya z^{-1}'in artan kuvvetlerinde saklanır. Örneğin;

$$H(z) = \frac{2z^2+3z+4}{z^2+5z+6} = \frac{2+3z^{-1}+4z^{-2}}{1+5z^{-1}+6z^{-2}}$$

için vektörler;

num = [2 3 4];
den = [1 5 6];

olarak tanımlanır.

C. Zaman Benzetimleri (Simulations)

Kullanılan komutlar:

recur
conv
dstep
dimpulse
filter

$$y[n] + \sum_{i=1}^{N} a_i y[n-i] = \sum_{i=0}^{M} b_i x[n-i]$$

özyineli ilişkisi ile tanımlanan bir sistem yanıtının hesaplanması için üç yöntem vardır :
İlk yöntem **recur** komutunu kullanır ve sıfır olmayan başlangıç durumları olduğunda yararlıdır. Fonksiyona girişler $a=[a_1\, a_2 \ldots a_N]$ ve $b=[b_0\, b_1 \ldots b_M]$ vektörlerinde saklanan a_i ve b_i katsayılarıdır, x ve y 'de başlangıç koşulları x0 = [x[n0-M], x[n0-M+1],...,x[n0-1]] ve y0 = [y[n0-N], y[n0-N+1],...,y[n0-1]]] vektörlerinde gösterilir ve çözüm için hesaplanması gereken zaman indisleri n0'ın vektördeki ilk elemanı gösterdiği n vektöründe belirtilir. **recur** komutunu kullanmak için,

y = recur(a,b,n,x,x0,y0);

yazılır. Çıkış y[n] elemanlı bir y vektörüdür; y'nin ilk elemanı zaman indeksi n0'a denk düşer. Bu fonksiyon

```
function y = recur(a,b,n,x,x0,y0);
%
% y = recur(a,b,n,x,x0,y0)
%  solves for y[n] from:
% y[n] + a1*y[n-1] + a2*y[n-2]... + an*y[n-N]
%   =  b0*x[n] + b1*x[n-1] + ... + bm*x[n-M]
%
% a, b, n, x, x0 and y0 are vectors
% a = [a1 a2 ... aN]
% b = [b0 b1 ... bM]
% n contains the time values for which the solution will be
computed
% y0 contains the initial conditions for y, in order,
%    i.e., y0 = [y[n0-N], y[n0-N+1], ...,y[n0-1]]
%    where n0 represents the first element of n
% x0 contains the initial conditions on x, in order
%    i.e., x0 = [x[n0-M],...,x[n0-1]]
% the output, y, has length(n)
%
N = length(a);
M = length(b)-1;
if length(y0) ~= N,
  error('Lengths of a and y0 must match')
end
if length(x0) ~= M,
  error('Length of x0 must match length of b-1')
end
y = [y0 zeros(1,length(n))];
x = [x0 x];
a1 = a(length(a):-1:1);              % reverses the elements in a
b1 = b(length(b):-1:1);
for i=N+1:N+length(n),
  y(i) = -a1*y(i-N:i-1)' + b1*x(i-N:i-N+M)';
end
y = y(N+1:N+length(n));
```

Örneğin; x[n] = u[n], başlangıç koşulları y[-1] = 2, y[-2] = 1, ve x[-1] = x[-2] = 0 olarak verilen,

y[n] - 0.6y[n-1] + 0.08y[n-2] = x[n-1]

şeklindeki sistem ele alınsın. n = 0, 1,...,10 için y[n] yanıtı;

```
a = [-0.6 0.08]; b = [0 1];
x0 = 0; y0 = [1 2];
n = 0:10;
x = ones(1,11);
y = recur(a,b,n,x,x0,y0);
```

yazılarak hesaplanabilir. y vektörü n = 0,1,...,10 için y[n] değerlerini içerir.

İkinci yöntem konvolüsyon kullanır ve y'deki başlangıç koşulları sıfır olduğunda kullanışlıdır. Bu sistem, ilk önce sistemin darbe yanıtı h[n]'nin bulunmasını, daha sonra Bölüm 6.A.'da anlatıldığı gibi h[n] ve x[n]'in konvolüsyonunu gerektirir. Örneğin; y[-1]=y[-2]=0 başlangıç koşullarıyla yukarıdaki sistemin tanımlandığı düşünülsün. Bu sistem için darbe yanıtı, $h[n] = 5[(0.4)^n - (0.2)^n]u[n]$ şeklindedir. y[n]'i hesaplamak için gereken komutlar;

```
n = 0:10;
x = ones(1,11);
h = 5*(0.4).^n - 5*(.02).^n;
y = conv(x,h);
y = y(1:length(n));
```

y vektörü n = 0,1,...,10 için y[n] değerlerini içermektedir. Vektör length(n) uzunluğuna getirilmelidir çünkü hem x[n] hem de h[n] sonsuz süreçli sinyallerdir. Sonsuz süreçli sinyallerin konvolüsyonu için Bölüm 6.A'ya bakılabilir.

Üçüncü yöntem, sistemin transfer fonksiyonunun bilinmesini gerektirir. **filter** komutu başlangıç koşullarına ve herhangi bir girişe yanıtı hesaplarken, **dstep** ve **dimpulse** komutları sırasıyla birim basamak ve birim darbe yanıtlarını hesaplamaktadır. Payda katsayıları $den = [1 a_1 a_2 \ldots a_N]$ olarak ve pay katsayıları $num = [b_0\ b_1 \ldots b_M\ 0 \ldots 0]$ olarak gösterilir. Örneğin; yukarıdaki sistem y[-1] = y[-2] = 0 başlangıç koşullarıyla verilsin. n=0,1,….,10 için basamak yanıtının hesaplanmasında;

```
n = 0:10;
num = [0 1 0]; den = [1 -0.6 0.08];
```

y = dstep(num,den,length(n));

komutları yazılır. Yanıt **stem** kullanılarak çizidirilebilir. Darbe yanıtının hesaplanmasında, yukarıdaki komutlarda sadece **dstep** yerine **dimpulse** kullanılır.

Herhangi bir girişe yanıtı hesaplamak için, giriş dizisi x vektöründe belirtilir. **y = filter(num,den,x);** komutu sistem yanıtını hesaplamak için kullanılır. Sistem sıfırdan farklı başlangıç koşullarına sahipse, başlangıç koşulları bir v0 vektöründe belirtilmelidir. N=M=1 olan birinci dereceden bir sistem için, z_i=[b_1x[-1]-ay[-1]] olarak tanımlanır. N=M=2 olan ikinci dereceden bir sistemde $z_i = [b_1x[-1] + b_2x[-2] - a_1y[-1] - a_2y[-2] \quad b_1x[-1] - a_2y[-1]]$. 'dır. Sıfırdan farklı başlangıç koşullu yanıtı hesaplamak için,

y = filter(num,den,x,zi);

yazılır. Örneğin; x[n] = u[n] girişiyle ve y[-1] = 2, y[-2] = 1 başlangıç koşullarıyla önceki sistemi ele alalım. y[n] aşağıdaki komutlarla hesaplanır:

n = 0:10; x = ones(1:11);
num = [0 1 0]; den = [1 -0.6 0.08];
zi = [0.6*2-0.08*1, -0.08*2];
y = filter(num,den,x,zi);

D. Frekans Yanıtı Çizimleri

Bir sistem için transfer fonksiyonundan DTFT (Ayrık Zaman Fourier Dönüşümü) **freqz** komutu kullanılarak hesaplanabilir. 'num' ve 'den'de transfer fonksiyonunun pay ve paydası tanımlanır.

[H,Omega] = freqz(num,den,n,'whole');

komutu, Omega vektöründe bulunan frekanslarda birim çember etrafındaki eşit aralıklı n nokta için DTFT hesaplamaktadır. H'nin genliği abs(H) ile ve fazı da angel(H) ile bulunur. Ω için değer kümesi oluşturulurken, istenilen frekanslar için bir Omega vektörü tanımlanır, örneğin Omega = -pi:2*pi/300:pi -π, π aralığında değişen 301 uzunluklu bir vektör belirtir. Bu frekanslardaki DFTF'yi elde edebilmek için;

H = freqz(num,den,Omega);

yazılır.

E. Sayısal Filtre Tasarımı

Kullanılan komutlar:

bilinear
butter
cheby1
hamming
hanning

IIR filtre tasarımının analog prototip yöntemi; ilk önce Bölüm 5.D'de belirtilen karakteristiklere sahip bir analog filtre tasarlanmasıyla, daha sonra filtrenin ayrık zaman etki alanına haritalandırılmasıyla gerçekleştirilebilir. Analog filtre H(s)'nin pay ve paydası num ve den vektörlerinde saklansın ve örnekleme periyodu T olsun. Sayısal filtre H(z)'nin pay ve paydası aşağıdaki komutla bulunabilir:

[numd,dend] = bilinear(num,den,1/T)

butter ve **cheby1** komutlarıyla da otomatik olarak analog filtre tasarlanmaktadır ve bu komutlar filtrenin ayrık zaman etki alanına haritalandırılmasında çiftdoğrusal dönüşüm kullanmaktadırlar. Alçak geçiren, yüksek geçiren, band geçiren ve band durduran filtreler de bu yöntem kullanılarak tasarlanabilir. Sayısal kesim frekansları belirli olmalıdır ve πı ile normalize edilmelidir. Analog Butterworth filtreye dayandırılan bir sayısal alçak geçiren filtre tasarımı için;

[num,den] = butter(n,Omegac)

komutu kullanılır. n kutup sayısını ve Omegac ($\Omega_c = \omega_c T/\pi$) normalize edilmiş kesim frekansını göstermektedir. Omegac kesim frekanslı yüksek geçiren filtre için;

[num,den] = butter(n,Omegac,'high')

yazılır.

Geçiş bandı Omega1'den Omega2'ye kadar olan (Omega = [Omega1,Omega2] olarak tanımlanır) band geçiren filtre tasarımı için;

[num,den] = butter(n,Omega)

komutu kullanılır.

Durdurma bandı Omega1'den Omega2'ye kadar olan (Omega = [Omega1,Omega2] olarak tanımlanır) band durduran filtre;

[num,den] = butter(n,Omega,'stop')

ile tasarlanabilir.

n. dereceden bir Type I Chebyshev filtre tasarımı ise butter ile aynı yöntem kullanılarak ancak sadece "butter" yerine "cheby1" yazılarak yapılabilir:

[num,den] = cheby1(n,Omegac); % Alçak geçiren filtre için
[num,den] = cheby1(n,Omegac,'high'); % Yüksek geçiren filtre için

Omega iki elemanlı ise;

[num,den] = cheby1(n,Omega); %Band geçiren filtre için
[num,den] = cheby1(n,Omega,'stop'); %Band durduran filtre için

FIR filtre tasarımında kullanılan pencereler;

w = boxcar(N) % dikdörtgen pencere
w = hamming(N)
w = hanning(N)

ile verilir.
Bu komutlar N uzunluklu FIR filtreli sayısal ideal bir filtrenin sonsuz darbe yanıtını kısaltmak için kullanılır.
İşaret İşleme Araç kutusu (Signal Processing Toolbox) aynı zamanda FIR filtrenin doğrudan hesaplanmasını sağlayan komutları içerir. N uzunluklu ve Omegac (πı ile normalize edilmiş) kesim frekanslı bir FIR filtre elde edebilmek için;

h = fir1(N-1,Omegac)

yazılır.

h vektörü h(1)'in h[0] değeri olduğu FIR darbe yanıtını içerir.

Omegac normalize kesim frekanslı ve N uzunluklu yüksek geçiren filtre;

h = fir1(N-1,Omegac,'high')

komutu kullanılarak tasarlanabilir.

Omega1'den Omega2'ye kadar geçiş bandı olan bir band geçiren filtre;

h = fir1(N-1,Omega)

yazılarak elde edilir (Omega = [Omega1,Omega2]). Omega1'den Omega2'ye kadar durdurma bandı olan bir band durduran filtre için;

h = fir1(N-1,Omega,'stop')

yazılır (Omega = [Omega1,Omega2]). **fir1** komutu Hamming pencereyi hatalı kullanır. Diğer pencereler 'hanning' veya 'boxcar' opsiyonunun eklenmesiyle elde edilebilir. Örneğin;

h = fir1(N-1,Omegac,'high',boxcar(N))

ile dikdörtgen pencere kullanılarak Omegac kesim frekanslı bir yüksek geçiren FIR filtre oluşturulur.

F. Sayısal Kontrol Tasarımı

Kullanılan komutlar:

bilinear
c2dm
hybrid

Bir analog G(s) denetleyici, çiftdoğrusal dönüşüm ya da basamak yanıtı eşleme yöntemi kullanılarak sayısal D(z) denetleyiciye dönüştürülebilir. G(s)'nin pay ve paydası num ve den'de gösterilir. D(z)'nin pay ve paydası çiftdoğrusal dönüşümden,

[numd,dend] = bilinear(num,den,1/T)

komutu kullanılarak bulunabilir. Burada T, örnekleme frekansıdır. Basamak değişimsiz yöntem kullanmak için;

[numd,dend] = c2dm(num,den,T,'zoh')

komutu yazılır.

Sayısal denetleyici D(z)'ye sahip bir sürekli zaman sistemi G(s) yanıtının simüle edilebilmesi için, **hybrid** komutu kullanılır. Sistemin pay ve payda katsayıları numG ve denG'de; denetleyicinin pay ve payda katsayıları numD ve denD'de; referans giriş işareti r'de ve örnekleme zamanı T'de saklanır. Zaman vektöründeki artışlar, örnekleme zamanının bir tamsayıyla bölümü olacak şekilde seçilmelidir, örneğin; $t = 0:b:T_{son}$ (bm=T'dir). Komut;

[y,ud] = hybrid(numG,denG,numD,denD,T,t,r);

şeklinde kullanılır. Komut çıkışları, sistem yanıtı y ve sisteme giriş olan kontrol işareti ud'dir. M-dosyası ayrık zaman kontrolü hesaplayan bir döngü içerir ve daha sonra sabit kontrollü T saniye için sürekli zamanlı sistemi simüle eder. Proses bir sonraki T saniye aralığı için tekrar eder.

G. Durum Uzay Gösterimi

Kullanılan komutlar:

dlsim
dstep
dimpulse

Sürekli zaman durum uzay gösteriminde kullanılan komutların çoğu aynı zamanda ayrık zaman durum uzayı için de kullanılmaktadır. Örneğin; **ss2tf**, **tf2ss**, ve **ss2ss** komutları Bölüm 5.F'de anlatılan sürekli zaman kullanımıyla aynı şekilde ayrık zaman için de kullanılır. Ayrık zamanda kullanılan **lsim** komutu;

[y,x] = dlsim(A,B,C,D,u,n);

şeklinde yazılır ve çıkış y'de, durumlar x'te, giriş u'da ve zaman indisi n'de saklanır.